



Data Formats in HEP Analyses

Jakob Blomer

ROOT I/O Workshop, CERN
June 12, 2017

- Focus on the “last mile” of publication creation
- Data sets are relatively small
 - $\mathcal{O}(10^7)$ events
 - Tens to hundreds of properties per event
 - Volume: gigabytes to terabytes
- Processing tends to escape central computing workflows and resources
- Number of reads \gg number of writes
- I/O bound: little calculation

Little data lock-in:

final data sets can (sometimes need to be) reproduced
e. g. starting from CMS MINIAOD, ATLAS xAOD

What we want from the data format

① Fast turn-around

"Let me quickly plot. . ."

- Tuning cuts (partial reading of the data set)
- Feeding into ML framework (full reading of the data set)

② Integration, unleashing the data

- Libraries for C++, Python
- Machine learning frameworks (e. g. TMVA, TensorFlow)
- Big Data schedulers (e. g. Spark)
- Analytic tool kits (e. g. ROOT, R, SciPy)

What we do not necessarily need:

- Plethora of tuning options (likely to be unused)

Universally efficient data layout and encoding is challenging given large spectrum of storage performance characteristics

- Latency:
10 ns RAM \rightarrow 100 μ s 3D XPoint \rightarrow 1 ms SSD \rightarrow 10 ms HDD, LAN
between fastest and slowest: $\times 100\,000$
- Throughput:
20 GB/s RAM \rightarrow 2 GB/s SSD, 3D XPoint \rightarrow 100 MB/s HDD, LAN
between fastest and slowest: $\times 200$

Additionally: uneven scaling of components, e. g.
memory size vs. network bandwidth in laptops



- Well-defined encoding, not simply a memory dump
- Self-describing, schema included
- Support for complex, nested data types (records, arrays)
- Preserving floating point precision
- Checksumming
- Possibly schema evolution

- ROOT:
 - Stream of serialized C++ objects in columnar layout
- Protobuf:
 - Not a file format per se but (de)-serialization of small records
 - Possible file format: `schema || [size record-blob]*`
- SQLite:
 - SQL database in a file
- HDF5:
 - Popular in the HPC universe
 - There are HEP machine learning data sets in HDF5
 - Many options, somewhat a file format toolkit
 - Hierarchical “data sets” that each contain a “data space” (n -dimensional array)
- Avro:
 - From the Apache Hadoop/Spark universe
 - JSON schema, row-wise binary storage
- Parquet:
 - From the Apache Hadoop/Spark universe
 - *column*-wise binary storage



“Fruit Fly” Data Set: The LHCb OpenData Sample

Starting point: “What if I had my data set in format X ?”

- 8.5 million run 1 events $B \rightarrow KKK$
- Flat n -tuple, 26 branches (mostly floating point numbers)
- 21 branches needed for the toy analysis
- 2.4 million events can be skipped because one of the kaon candidates is flagged as a muon
- Single-threaded

On the simple end of the spectrum,
helps to understand performance base case

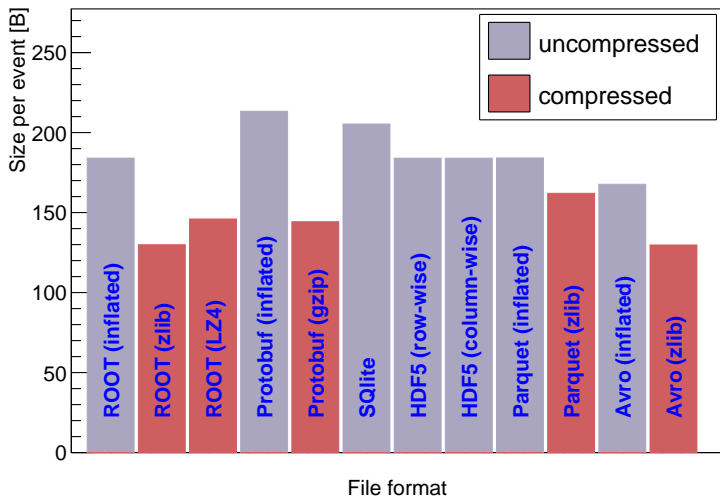
Links

<https://github.com/lhcb/opendata-project>

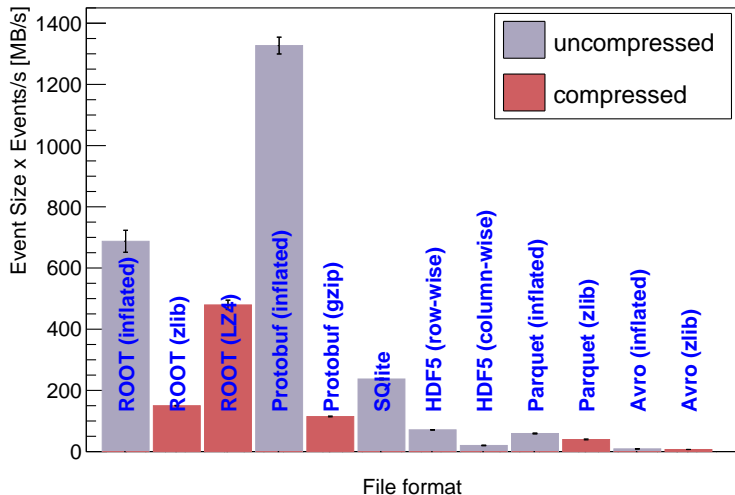
<https://github.com/lhcb/opendata-project/blob/master/Background-Information-Notebooks/EventData.ipynb>

<https://github.com/jblomer/iotools/tree/root-workshop>

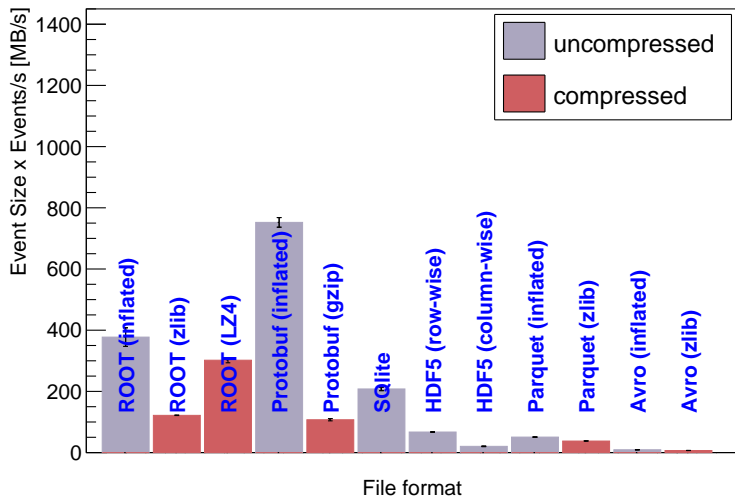
Data size LHCb OpenData



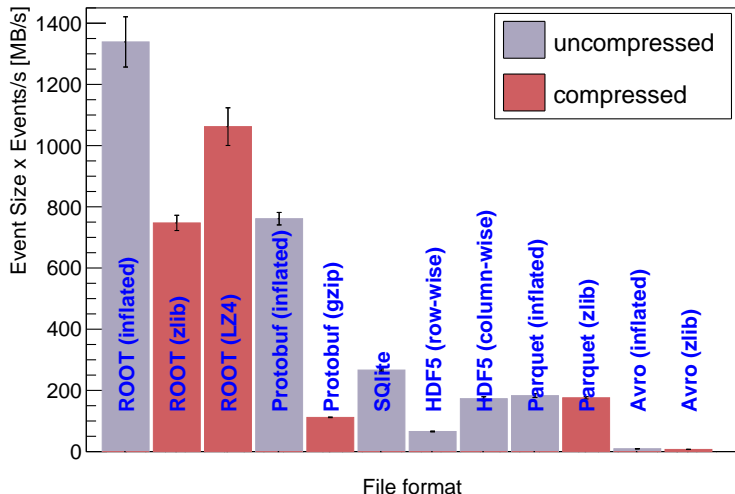
READ throughput LHCb OpenData, warm cache



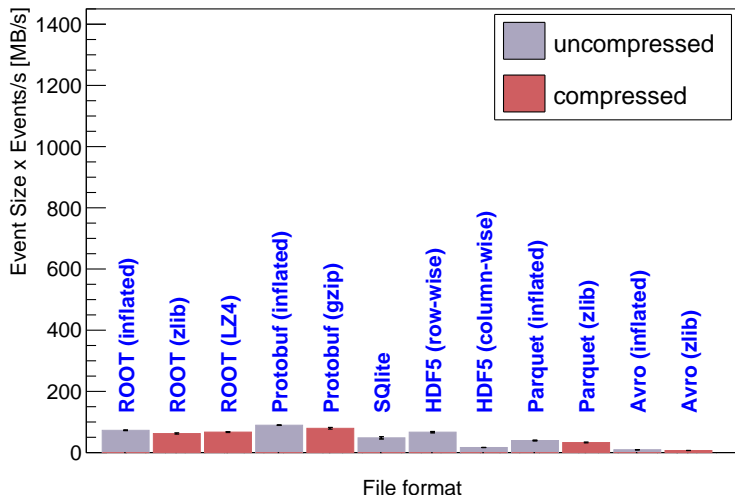
READ throughput LHCb OpenData, SSD cold cache

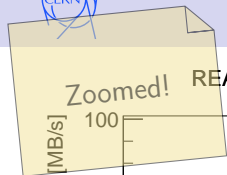


PLOT 2 VARIABLES throughput LHCb OpenData, SSD cold cache

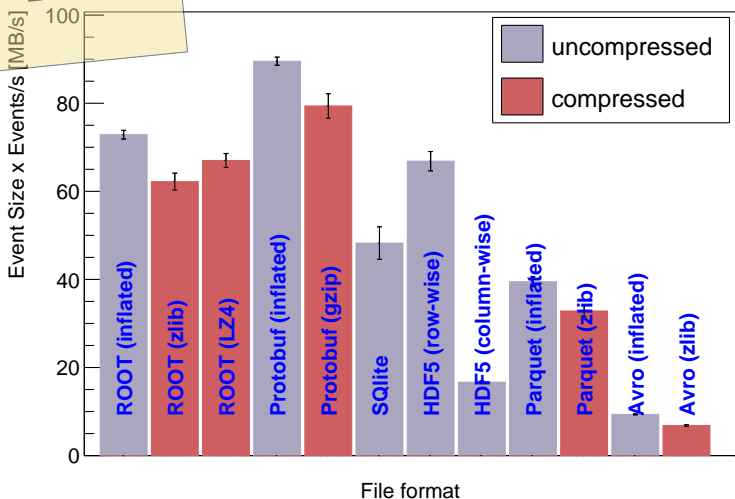


READ throughput LHCb OpenData, HDD cold cache

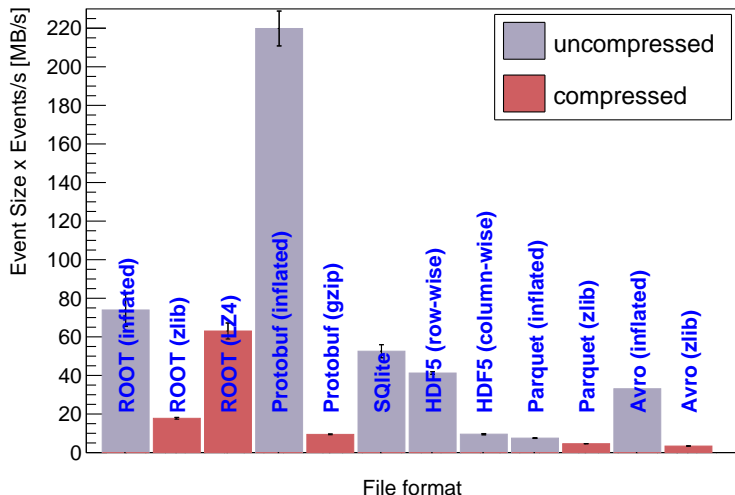




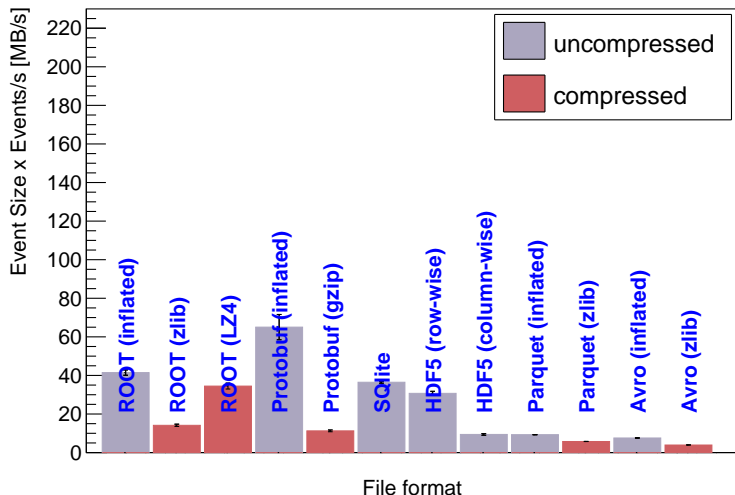
READ throughput LHCb OpenData, HDD cold cache



WRITE throughput LHCb OpenData, SSD



WRITE throughput LHCb OpenData, HDD



This is an ongoing study. I'm happy to add more file formats and data sets.

Thoughts on the numbers:

- On fast storage, performance is dominated by (de-)serialization
- We might want to change the compression default on fast storage (or aim for hardware acceleration for compression algorithms)

A repository of **archetype analyses** would be very helpful.
Prepared data sets (not necessarily real data) *and*
a description of the analysis routine.

We can benefit a great lot from **integration work**.
That can mean teaching ROOT other data formats *or*
teaching 3rd party software the ROOT format.

Backup

Hardware	Type
CPU	i7-6820HQ @ 2.7 GHz
Memory	2×16 GB DDR4 2133 MHz
SSD (flash)	1 TB Toshiba XG3 PCIe
HDD (spinning)	Western Digital WD20NMVW

Library	Version
ROOT	6.08/06
protobuf	3.2.1
sqlite	3.18.0
hdf5	1.10.0_patch1
avro-c	1.8.1
parquet-cpp	1.0.0

Operating system: Linux 4.10, glibc 2.25, gcc 6.3.1